|  |  |
| --- | --- |
| CUS 1159 – Advanced C#. | Sean Munson |
| Project Proposal; | March 4 2010 |

**Data Service Composition Tool**

Modern software design can sometimes be require that database design be disconnected from business-layer manipulations. On the other hand, frequent changes can put a heavy load on development resources and can expose a product to transcription or synchronization errors. How can a development group keep their software up-to-date and when underlying database changes may require destruction or construction of whole object libraries of data objects ?

An Automated tool that, in a regular, reliable way, provided strongly typed object class library construction from the present data structures, would substantially mitigate this problem. It also would be a way to regularly implement some other features, such as range-checking and techniques to preclude SQL injection attacks. In the future, one could see such works actually making it possible to reduce the overall exposure of the database, using limited, generated access points for Data Access.

**Proposal :**

I propose to develop a simple version of this kind of product, designed to allow for future expansion to multiple future possible database products and multiple possible Language supports. It will allow a development team to generate and persist augmentations to a standard set of features.

**Overivew:** Users will be able to configure the tool to connect to a database, giving the program the ‘correct’ connection string. They will be permitted to select one, all or no Tables & Views for production to Data Objects. The user will be able to select which stock methods in which objects will be produced. They will be able to save this configuration for later modification. They will then choose a language, and a directory, and the program will generate the data classes, a connection management class and a factory class.

**Example :** Bob wants to generate a dataservice project for his existing SQL database. He opens the tool, and enters the connection string for the SQL server. He then selects the ten tables and two views which he needs to access in his application, and selects SELECT, UPDATE, INSERT operations for the tables, and the SELECT operation for the views. He then saves his work, so he can alter it in the future as the project expands. He Selects C# and c:\projects\MyDataservice as his output parameters and the program generates 14 files : one management, one factory, and 12 data access classes.

Bob can now use these in a project, or (if the project already existed) he can recompile his project to accommodate any updates to the dataset that have occurred. If the database is altered, or more qualifications are generated , he can revise the build document and re-generate the dataservice. The Consistent derivation of classes and properties will allow for minimal impact to code.

**Interfaces:** The product will be designed to be implemented in a number of segments, each distinguished by their location in relations to a critical component interface:

*‘Native’ application classes* : Classes used to manipulate the set of objects to be generated, and to proffer the options for generating them.

*DataProvider* : Classes that Implement the IDataProvider Interface will provide bare-bones connectivity to databases, in particular, getting lists of tables and information on table structures.

*Xmlserializer* : used to dynamically author and read the build document used to save work between application sessions.

*QueryBuilder* : These components are designed to create Database-specific SQL or Stored procedure code for optimized performance and to support multiple database types.

ProjectComposer : Project composers : Will generate the code for the selected language.

**Diagram :**

